Figure 1

Figure 2

Figure 3

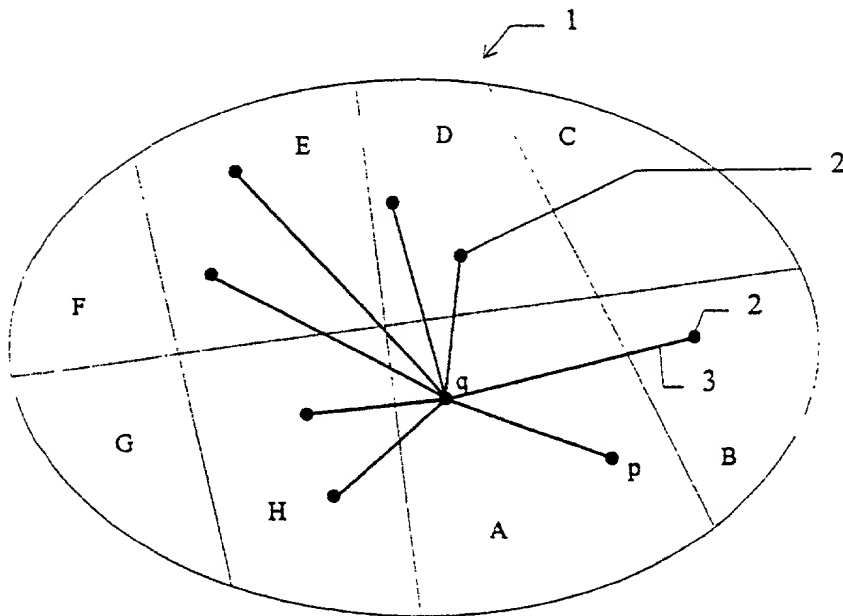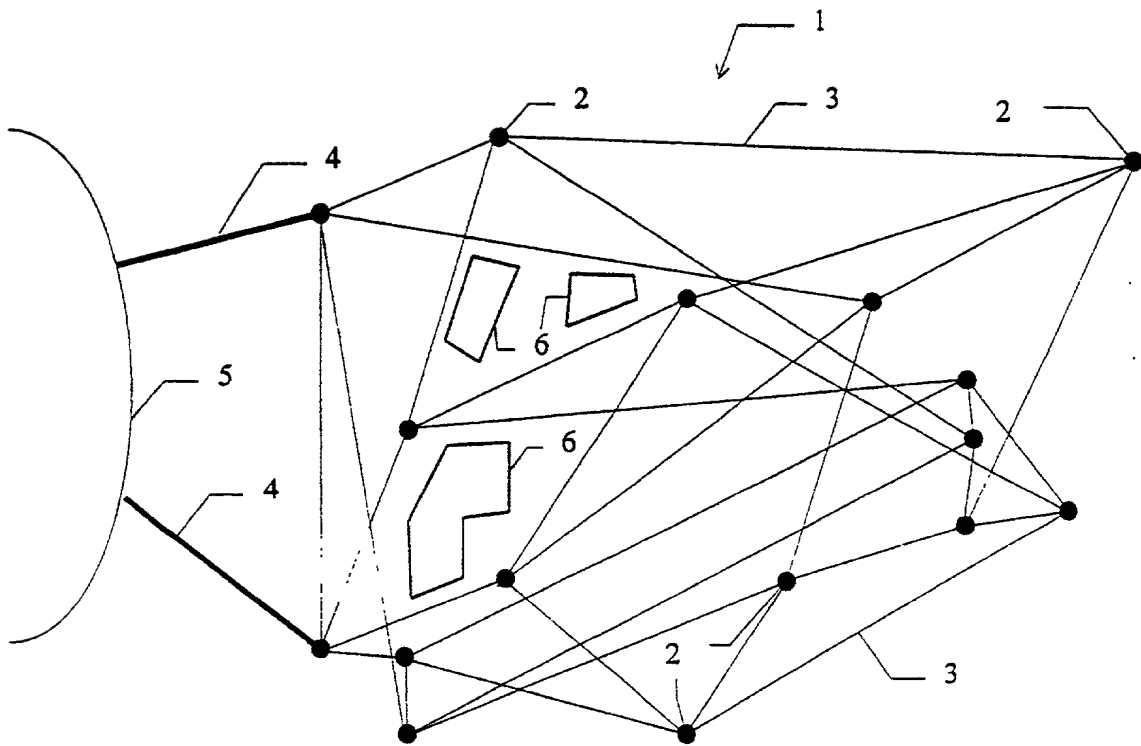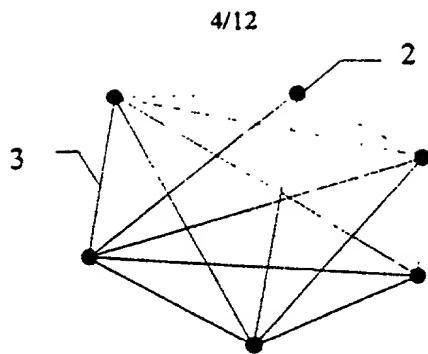Figure 4

Figure 5

Figure 6
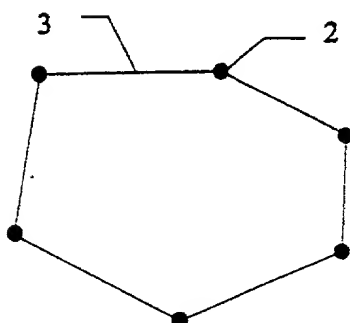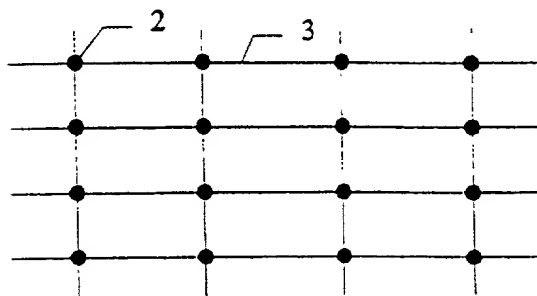
Figure 7
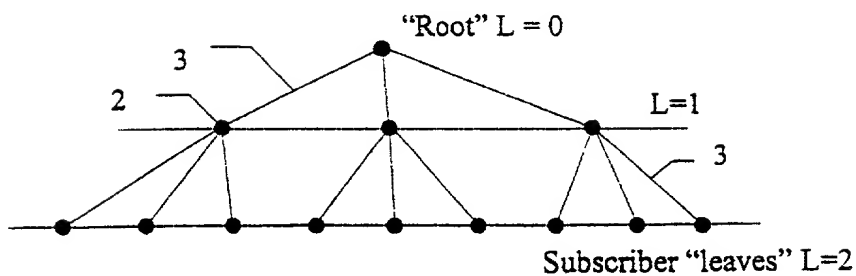
Figure 8

Figure 9

Figure 10

Figure 11

	A0	A1	A2	A3	A4	A5	A6	A7
T0	1	-	-	-	-	-	-	-
T1	-	1	-	-	-	-	-	-
T2	-	-	1	-	-	-	-	-
T3	-	-	-	1	-	-	-	-
T4	-	-	-	-	1	-	-	-
T5	-	-	-	-	-	1	-	-
T6	-	-	-	-	-	-	1	-
T7	-	-	-	-	-	-	-	1

Figure 12A

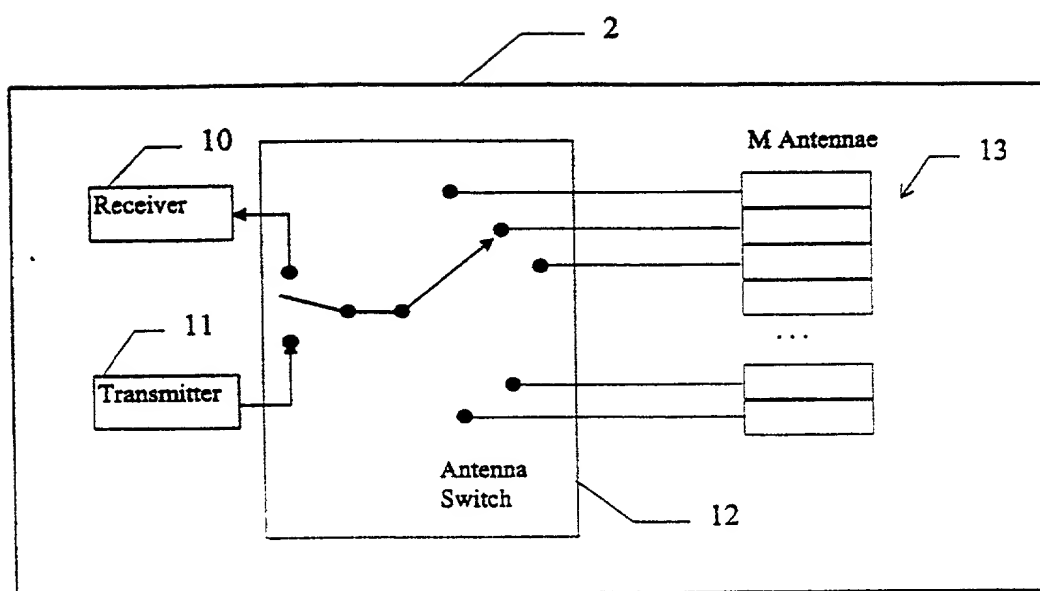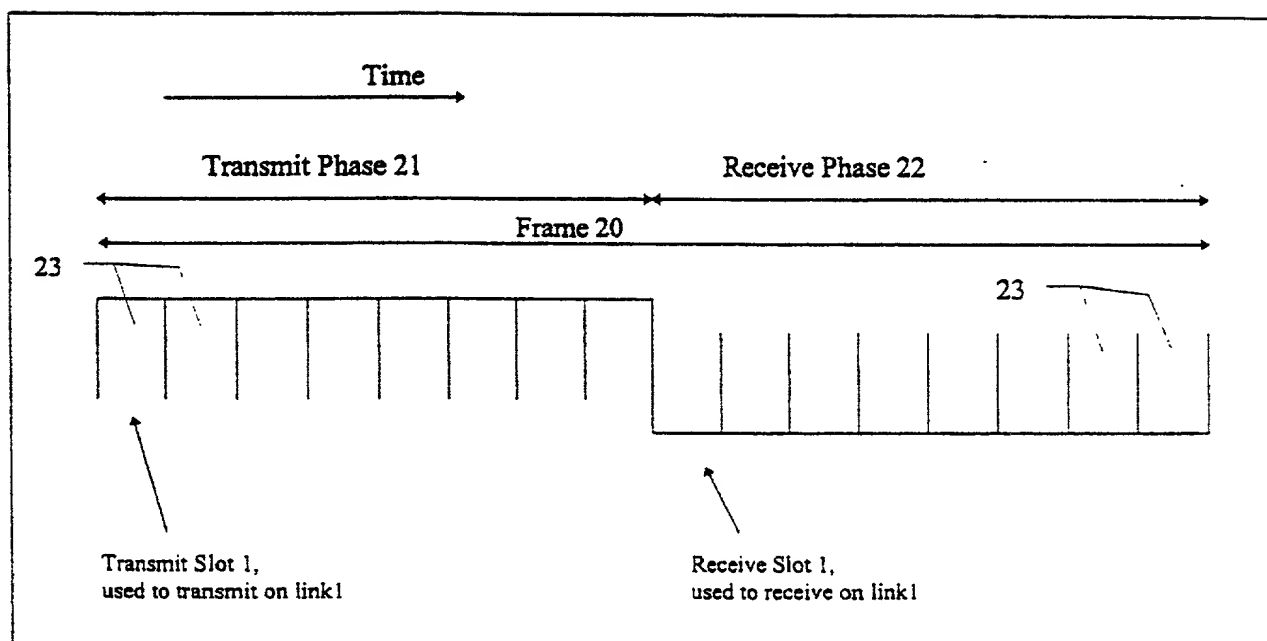|	A0	A1	A2	A3	A4	A5	A6	A7
T0	1	-	-	-	-	-	-	-
T1	1	-	-	-	-	-	-	-
T2	-	1	-	-	-	-	-	-
T3	-	-	1	-	-	-	-	-
T4	-	-	-	-	1	-	-	-
T5	-	-	-	-	1	-	-	-
T6	-	-	-	-	1	-	-	-
T7	-	-	-	-	-	-	-	1

Figure 12B

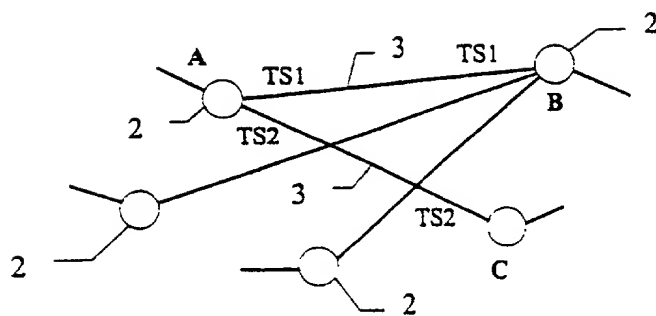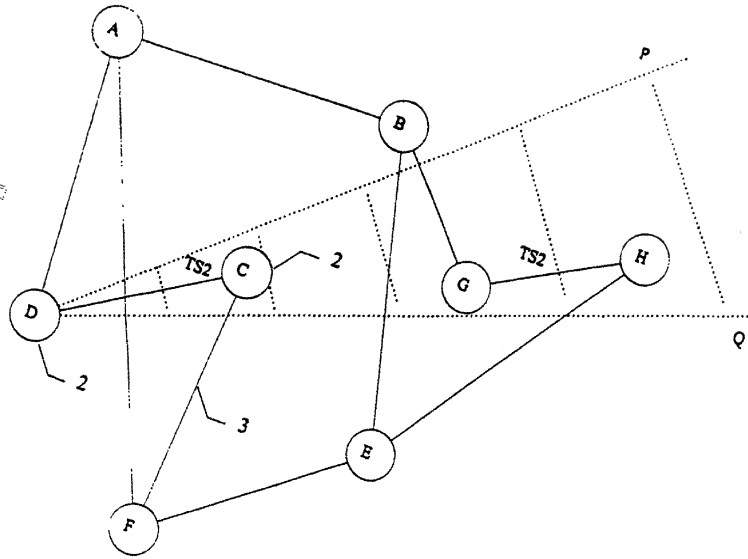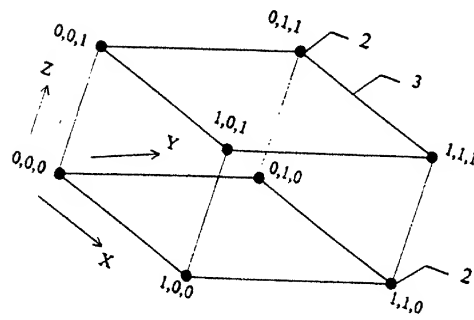|	A0	A1	A2	A3	A4	A5	A6	A7
T0	-	-	-	-	1	-	-	-
T1	-	-	-	-	1	-	-	-
T2	-	-	-	-	1	-	-	-
T3	-	-	-	-	1	-	-	-
T4	-	-	-	-	1	-	-	-
T5	-	-	-	-	1	-	-	-
T6	-	-	-	-	1	-	-	-
T7	-	-	-	-	1	-	-	-

Figure 12C

Figure 13

Figure 14

Figure 15

```
Begin RoutePacket ()

Define L of type Address;
Define next_channel, input_channel of type Channel;
Define msg of type Message;
Import messageQueue of type Queue;

forever
        if length of (messageQueue) > O then
                dequeueMessage (messageQueue, (msg, input_channel));

                if msg.status = Returned then
                        -- this message has been returned.
                        handleReturnedMessage (msg, input_channel);
                else
                        -- Pass message on to next node.
                        set L := msg.L

                        if L = my_L then
                        -- Packet has arrived - terminates here.
                                ProcessCell (msg.cell);
                        else
                                set next_channel := decideNextChannel (L,my_L);

                                if next_channel = NoBestChannel then
                                        set msg.status := Returned;
                                        set next_channel := input_channel;
                                end if
                                SendPacketToChannel (msg, next_channel);
                        end if
                end if
        end if
end

end RoutePacket.
```

Figure 16

Begin decideNextChannel (Address L, Address my_L) of type Channel

    define hop of type Address;
    define weightedChannelSum, sum, weightedChannel of type Real Number;
    define bestChannel of type Channel;
    define j, unuseableChannels of type Integer;

    set hop := L - my_L;
    set weightedChannelSum := 0.0;
    set sum := 0.0;

    for j := 0 to Length of(hop)
    {
        if ChannelUtilisation (j) > MaximumChannelUtilisation then
            set unuseableChannels := unuseableChannels + 1;
        end if
        set weightedChannelSum:= weightedChannelSum+hop[j]*j/ChannelUtilisation (j);
        set sum := sum + hop [j] / ChannelUtilisation [j];
    }

    if unuseableChannels = ActiveChannels then
    -- message cannot be forwarded from this node must be handed back to sender.
        return NoBestChannel;
    end if
    set weightedChannel := weightedChannelSum / sum;
    set bestChannel := MapWeightedChannelToBestChannel (weightedChannel);
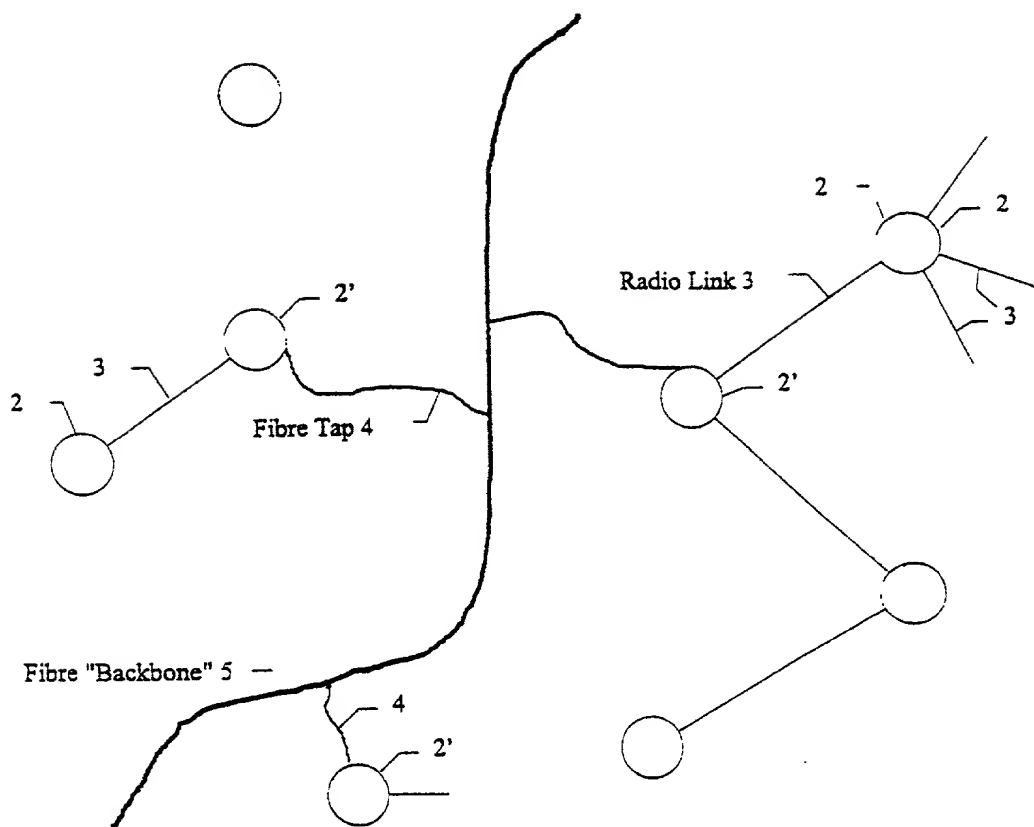
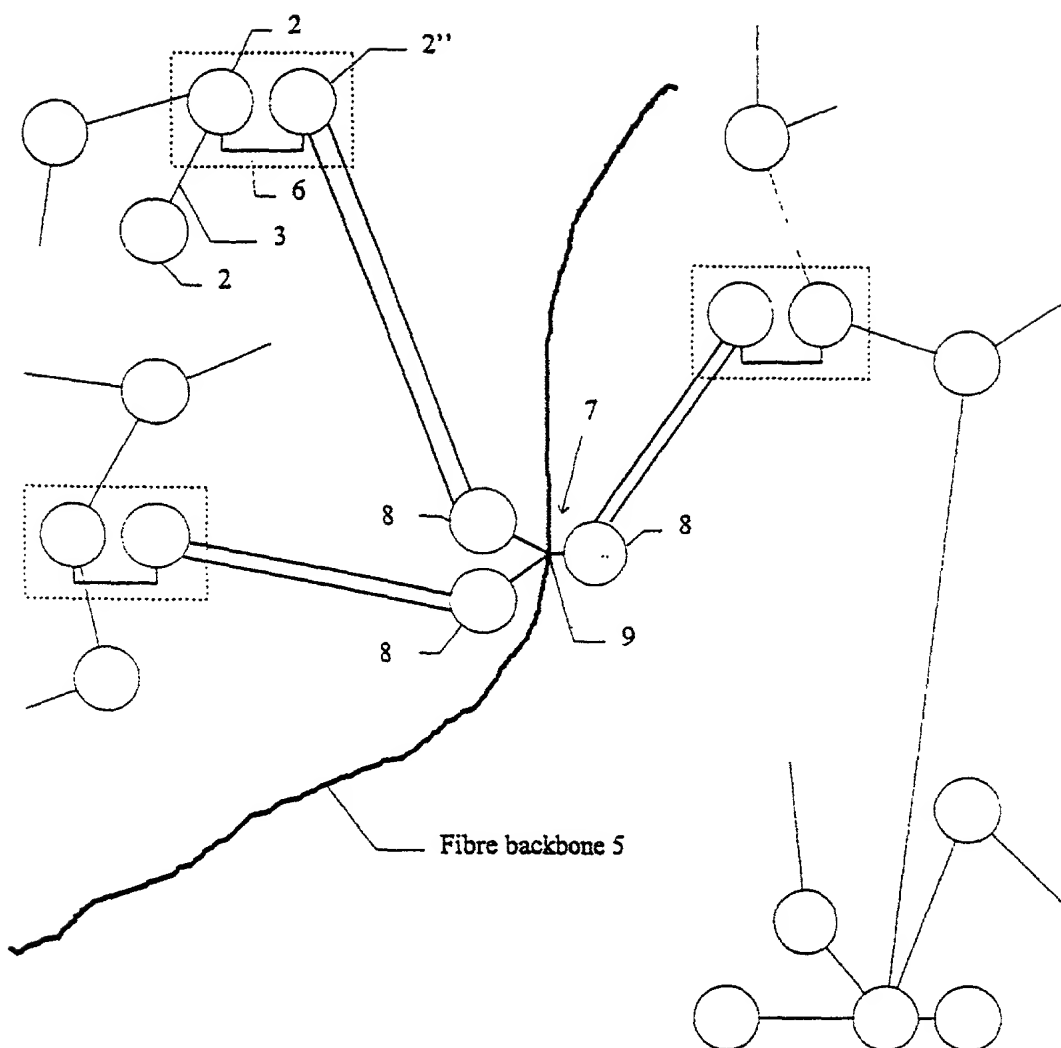   return bestChannel;
end decideNextChannel

Figure 17

Figure 18

Figure 19